An ID-based authentication scheme for the IEEE 802.11s mesh network

Aymen Boudguiga, Maryline Laurent

An ID-based Authentication Scheme For the IEEE 802.11s Mesh Network

Aymen Boudguiga, Maryline Laurent
Institut TELECOM, TELECOM SudParis, CNRS Samovar UMR 5157
9 rue Charles Fourier, 91011 Evry, France
Email: {Aymen.Boudguiga, Maryline.Laurent}@it-sudparis.eu

***Abstract*—Nowadays authentication in Wireless Mesh Networks (WMN) refers to the 802.1X authentication methods or a Preshared key authentication, and makes use of certificates or shared secrets. In wireless environments, management of certificates is disadvantageous. Certificates require deploying a Public Key Infrastructure (PKI) and Certification Authorities (CA) and they require defining a certificate management policy to control the generation, transmission and revocation of certificates. Management of certificates is a cumbersome task and does not match the limited (power and memory) resources available at wireless nodes. Moreover it does not match the non permanent connectivity to CA.**
In this paper, we propose an ID-based method, as an alternative to the PKI, to provide nodes with private and public keys, and we present an authentication scheme that uses the ID-based cryptographic concepts. As illustrated in the paper, the authentication scheme is shown as suitable to the WMN networks.

I. Introduction

Deployment of Wireless Mesh Networks (WMNs) is growing as they serve advantageously to enlarge wired backbones (mostly MANs), especially in country side areas where the cost of introducing a wired network to few costumers is very expensive. WMNs can also be used as home networks to interconnect all the home devices together with no constraints for the traffic to go through a hub.

WMNs are constructed as community networks. That is, a community peer to peer network is built within the WMN, and it fully supports local traffic transfering between stations of the community thanks to some multi-hop routing protocols. As such, the local traffic is not routed through the Internet (Internet routers), and direct advantages are lightened load of routers and increased link bandwidth.

As mesh networks are spreading quickly, providing a reliable authentication service is becoming compulsory. Most of the current authentication solutions rely on the 802.1X standard [1] and the Extensible Authentication Protocol (EAP) [2]. They refer either to a public/private key pair or to a secret shared between the two authenticating stations. Classical asymmetric cryptography assumes the existence of a CA to manage public key certificates as part of a Public Key Infrastructure (PKI) where a security policy is defined and enforced. Managing certificates is too much cumbersome in mobile networks like mesh and ad-hoc networks. As such alternative authentication must be investigated.

In this paper, we present a new authentication mechanism adapted to WMNs and using ID-based cryptography. ID-based cryptography considers the station identity as its public key, and makes it possible to derive a corresponding private key. This derivation function, as well as the secure transmission of the private key to its owner are performed by the Private Key Generator (PKG), also called Trust Agent (TA). Note that ID-based cryptography requires lightweight implementations at the client level. Compared to PKI certificate management, it does not need any special space for certificate storage, and the key revocation operation is easier. Key revocation in ID-based cryptography is bound to a validity period which is defined by the PKG. Please refer to the article [3] for a good comparison between PKI and ID-based cryptography.

Our article is organized as follows. First, the ID-based cryptography and signature mechanisms are introduced. Then, several mesh network architectures are described, and our ID-based authentication scheme is presented enriched with some possible extensions to maintain the security level throughout the sessions. Finally, details about how our scheme can apply to the 802.11s network environment are given.

II. ID-based Signatures

ID-based signature was initially introduced by A. Shamir to provide entities with public/private key pairs with no need of certificates, CA and PKI. Each entity uses a pair of its identifiers as its public key [4]. These identifiers have to be unique. In addition, the private key generation is assigned to a special entity which is called Private Key Generator (PKG). As such, before accessing the network, every entity has to contact the PKG in order to get a smart card which contains the private key of the entity. This private key is computed so it is bound to the public key of the entity.

During the last decade, the ID-based cryptography was enhanced by the use of the Elliptic Curve Cryptography (ECC). As a consequence, new ID-based signature schemes appeared and differed from Shamir's method in that the PKG does not rely on smart card to store the private key

and the ciphering information. The main objectives of the PKG are supporting the private key generation and the secure key distribution.
In the next two sections, we present two ID-based signature schemes using the ECC.

A. K.G. Paterson ID-based Signature Scheme

K.G. Paterson proposed in 2002 an ID-based signature scheme using ECC [5]. The success of this signature scheme depends on the success of the private key derivation phase that takes place when a node joins the network. Each node has to provide the PKG with the identity ID that it intends to use for its private key computation. The PKG then derives the node's private key using some parameters.
Paterson defines the following parameters to be used during the execution of his ID-based signature scheme. Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order q. G_1 is a subgroup of the group of points of an Elliptic Curve (EC) over a finite field and G_2 is a subgroup of a multiplicative group of a related finite field. Paterson also assumes the existence of a bilinear map $\hat{e}$ from $G_1 \times G_1$ in G_2 and the existence of an element $P \in G_1$ such that $\hat{e}(P,P) \neq 1_{G_2}$. The point P is used to compute another point P_{pub} which is presented in the following. Furthermore, Paterson supposes the presence of three hash functions H_1, H_2 and H_3 such that: $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : G_1 \rightarrow \mathbb{Z}_q^*$. The aforementioned parameters are known as the *public elements*. These public elements are distributed by the PKG to the network users because they are required during the public key derivation and the signature operation.
The key derivation operation starts when the PKG receives the ID of the node that is requesting a private key. First, the PKG computes the user's public key as $Pub_{ID} = H_1(ID)$. Then, the PKG generates the corresponding private key using a local secret value $s \in \mathbb{Z}_q^*$. Note that the private key is computed as: $Priv_{ID} = s \cdot Pub_{ID}$. The secret value s is also used for P_{pub} derivation from P: $P_{pub} = s \cdot P$.
In order to compute the signature of a message M, a user generates a secret random $k \in \mathbb{Z}_q^*$ and computes its signature as the pair $(R,S) \in G_1 \times G_1$ where: $R = k \cdot P, S = k^{-1}(H_2(M) \cdot P + H_3(R) \cdot Priv_{ID})$.
The signature verifier has just to compare $\hat{e}(R,S)$ to $\hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{ID})^{H_3(R)}$. The two values must be equal in order to consider the signature as valid.

B. F. Hess ID-based Signature Scheme

F. Hess presented its ID-based signature scheme in 2003 [6]. His signature scheme keeps the previous public parameters definition but it replaces H_2 and H_3 by a new hash function that we denote as $H_4 : \{0,1\}^* \times G_2 \rightarrow \mathbb{Z}_q^*$. In order to sign a message M, the user chooses an arbitrary point $P_1 \in {G_1}^*$ and a random $k \in \mathbb{Z}_q^*$. In addition, it executes the following steps:

1) $r = \hat{e}(P_1, P)^k$
2) $v = H_4(M, r)$
3) $U = v \cdot Priv_{ID} + k \cdot P_1$

The signature is formed by the pair $(U,v) \in G_1 \times \mathbb{Z}_q^*$.
The signature verifier then has to compute:

1) $r = \hat{e}(U,P) \cdot \hat{e}(Pub_{ID}, -P_{pub})^v$
2) The signature is accepted if and only if $v = H_4(M,r)$

Integration of this ID-based signature scheme into an EAP authentication method was submitted by Wenju et al. in 2009 [7]. The proposed scheme defines a dedicated server to implement the role of the PKG. This server verifies the uniqueness of the node's identity before generating its private key. Wenju et al. make the strong assumption that the channel between the PKG and the node requesting the private key is secure. In addition, they assume that the PKG is sending the Authentication Server (AS) identity to the node. At the end of the key derivation phase, the node gets its private key, the *public elements* and the identity of the AS which is responsible for authenticating all the supplicant stations. The node is now requested to authenticate itself to the AS using the Hess ID-based signature scheme before communicating with another station in the network.

III. Wireless Mesh Network Architecture

Mesh networks are actually expanding due to their easy deployment and low management cost. Many standardization groups have been attracted by these networks and started developing standards and protocols adapted to the mesh network requirements. IEEE is developing its own mesh standards in accordance to its existing wireless technologies. IEEE 802.11s is the WLAN based mesh standard which defines a new routing protocol called Hybrid Wireless Mesh Protocol (HWMP) and a new routing metric called Airtime Metric [8]. IEEE also defines 802.16A and 802.15.5, respectively as the WMAN and WPAN mesh based networks. Some proprietary mesh solutions and testbeds are today deployed such as Tropos and MIT roofnet mesh networks. They use generally the 802.11 technologies enhanced with some modifications in order to provide multihop routing protocols.
We next present the existing mesh architectures, as well as the IEEE 802.11s mesh architecture to which our authentication scheme is applied in section VI.
WMN architectures can be divided into three groups:

- *Infrastructure mesh:* is formed by special mesh nodes (or points) called routers. Mesh routers are quasi-static (with limited mobility), and form a wireless backbone. They serve to route traffic between customers and they act as gateways in case traffic is addressed to some external networks (Internet...). The provided wireless backbone is very efficient in countryside areas where the deployment of a wired infrastructure is time consuming and very expensive. It brings a low-cost wireless infrastructure which is easy to deploy, and highly scalable.
- *Client mesh:* is composed of mobile nodes called clients. These mesh points have no gateway capability.

A client is only able to forward the traffic between its neighbors, but it can not act as a gateway. The client mesh has the same characteristics as an ad-hoc network.

- *Hybrid mesh:* is a combination of the two previous mesh networks (Figure 1). The client mesh is connected through a mesh router to the backbone. Two mesh clients from different wireless technologies can be interconnected through the backbone. For example, a mesh client is able to communicate with a Wi-Fi client.

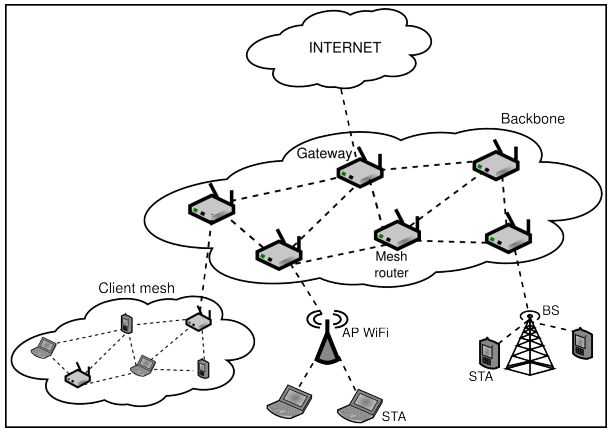

Fig. 1. Hybrid mesh.

The IEEE 802.11s architecture is based on the IEEE 802.11 architecture which is formed by Stations (STAs), Access Points (APs) and a Distribution System (DS) [9]. In 802.11, every AP offers connectivity to a number of STAs. The group formed by the AP and the STAs is called the Basic Service Set (BSS). The DS serves to interconnect different BSSs through a wired network.

The IEEE 802.11s standard introduces modifications to the 802.11 architecture. First, the wired DS is replaced by a backbone composed of a set of Mesh Points (MPs) (called also Mesh Routers or Mesh STA - MSTA). These wireless MPs provide multi-hop paths and peer to peer communications to the Mesh APs (MAPs). A MAP has the same capability as a traditional AP combined with mesh function. Mesh points which offer connectivity to external networks (either 802 LANs or layer 3 networks) are called Mesh Portals (MPP) or Gateways. All these components (MPs, MAPs and MPPs) form the Mesh BSS (MBSS).

The 802.11s architecture defines new functions for some mesh STAs in order to provide security services such as station authentication and key derivation. The first function is the Mesh Authenticator (MA) which acts as a pass-through server during the authentication phase for the supplicant mesh STA to authenticate to the network Authentication Server (AS). The functions of "supplicant", "authenticator" and "authenticator server" are inherited from the EAP standard [2]. In addition, the standard defines the Mesh Key Distributor (MKD) as the entity that derives the keys needed for the upcoming 4-Way Handshake that occurs between the MA and the supplicant mesh STA. The MKDs serve to distribute the key derivation function that was used to be performed by the AS (in IEEE 802 networks). Each MA must be connected to an MKD because the latter is going to provide the former with the key needed to secure the communication with the supplicant. The standard assumes that there is a security association between the different authentication entities: AS-MKD, MKD-MA. Moreover, the security association between the MA and the supplicant mesh STA is created after a successful authentication of the supplicant STA by the AS. Figure 2 illustrates the different entities that are used during the 802.11s station authentication and key derivation operations. Moreover, these entities do serve our authentication scheme presented in section VI.

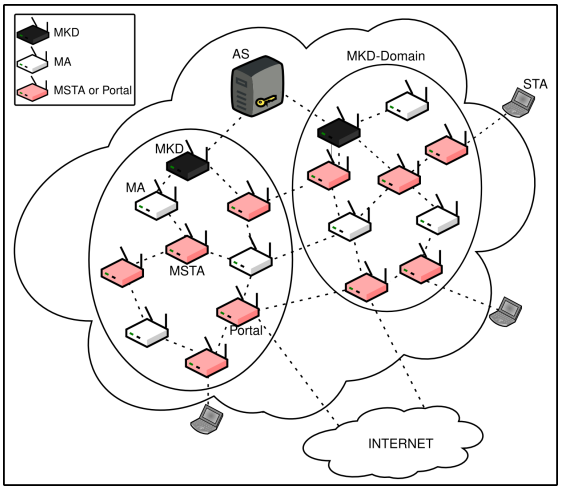

Fig. 2. IEEE 802.11s security components.

IV. ID-based Authentication Scheme

Nowadays, most of the authentication schemes that are being proposed for wireless networks uses the 802.1X standard [1]. The authentication method proposed by this standard are either based on the verification of a secret shared between two STAs or a signature mechanism that lies on the use of certificates in order to authenticate the public/private key pair used for signing. Management of public/private key requires deploying CAs to control the generation, revocation and duration of certificates. This is disadvantageous in wireless environments, such as ad-hoc and mesh networks, where stations may have some power and memory constraints and CA reachability is not guaranteed.

To mitigate these disadvantages while keeping usage of public/private key, we propose a new ID-based authentication scheme which permits a STA to authenticate itself to an AS when it initially joins the network. The AS is assumed to act as the Private Key Generator (PKG) and it serves to derive the private key of STA. Merging AS and PKG helps decreasing the traffic load for the STA to get its private key from the PKG. In addition, we assume that the AS and the authenticating STA share a secret value (i.e.

a password) that they use with ID-based cryptography to mutually authenticate.
After a successful initial authentication, the STA gets its private key corresponding to its ID-based public key, as described in section IV-A. For subsequent authentications, the STA uses a signature mechanism in order to authenticate itself with another STA as illustrated in section V-B.

A. STA Initial Authentication

The initial authentication occurs when a STA joins the network for the first time or after being disconnected for a while. To perform an ID-based authentication, the STA must first get the *public elements* that are published by the AS (acting in our proposal as a PKG). Then STA authenticates itself to the AS using a preshared secret. The secret may be a password, and is noted as *pwd* in our authentication scheme.
Note that the *public elements* are defined according to the selected ID-based signature scheme. If K.G. Paterson signature scheme is in use, the *public elements* are G_1, G_2, H_1, H_2, H_3, P and P_{pub}. If F. Hess signature scheme is used, the *public elements* are G_1, G_2, H_1, H_4, P and P_{pub}. These parameters are used either for signature computation or public key derivation.
Likely to RFC about EAP [2], we next call "supplicant" the STA requesting the authentication, and "authenticator (A)" one of its one-hop neighbors acting as a pass-through server to the Authentication Server (AS). Our proposed ID-based authentication scheme is illustrated in Figure 3.
When a STA joins the network, it listens to the *Beacons*

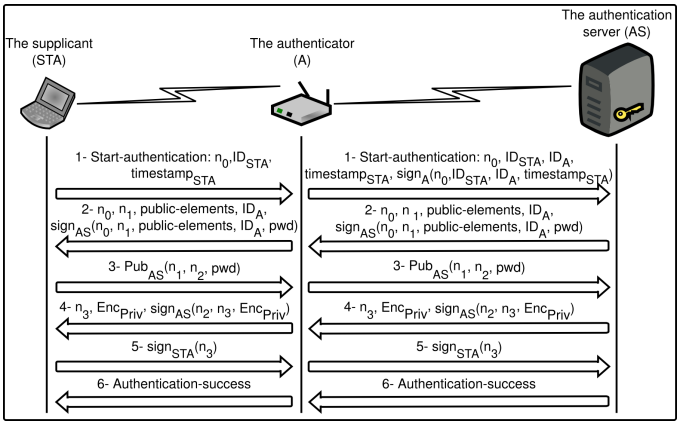

Fig. 3. ID-based authentication scheme.

from its neighbors and chooses one of them as authenticator. The STA then adjusts its internal clock to the authenticator's one. It starts the authentication mechanism by sending the *Start-authentication* message to the authenticator. The STA includes a random number n_0, a timestamp and its identity in the message. The identity (ID_{STA}) corresponds to the identity registered in the AS with *pwd* to authenticate the supplicant STA. In addition, this ID_{STA} represents the identity that serve to compute the public key of the supplicant.
When the authenticator (A) receives this message 1, it appends its identity ID_A to the message. Then, it signs the content of the message with its private key $Priv_A$. It forwards message 1 to the AS. Likely, the authenticator is acting as a pass-through server between the supplicant STA and the AS for the upcoming messages.
Upon receiving message 1, the AS verifies the timestamp value and the signature of the authenticator (A). The AS looks for the *pwd* corresponding to the received STA ID_{STA} in its password database. The AS then generates the message 2 which contains a new random number n_1, the received n_0, the identity of the authenticator, the *public elements* and a message signature. That signature is computed over the concatenation of n_0, n_1, the authenticator's identity, the *public elements* and the *pwd* shared with the supplicant STA.
After receiving message 2, STA checks the n_0 value to prove that message 2 is a response to its message 1 request. In addition, the supplicant checks the identity of the authenticator to guarantee that the authenticator is already authenticated by the AS. Then, the supplicant STA constructs the signed message by concatenating the received random numbers, the identity of the authenticator and the *public elements* to the secret *pwd* shared with the AS. Before the signature verification, the supplicant STA needs the AS public key (Pub_{AS}). In order to recover this Pub_{AS}, the supplicant executes the following operation: $Pub_{AS} = H_1(ID_{AS})$. The hash function H_1 is one of the *public elements* and ID_{AS} is the AS identity. As a result of Pub_{AS} computation, the STA becomes able to verify the AS signature of the reconstructed message.
If the signature verification fails, that means that either n_1 or the *public elements* are wrong or have been modified during the transfer of message 2. Receiving a wrong value of the *public elements* implies that the Pub_{AS} computed by the supplicant STA is not a valid one. The signature verification failure can also result from the usage of an invalid *pwd*. The supplicant STA does not successfully authenticate the AS unless a valid signature is received. The authentication process has to be stopped when the signature verification fails.
If the signature verification succeeds, the supplicant STA generates message 3. The supplicant STA picks a random number n_2 and verifies that q does not divide n_2, where q is the prime order of G_1 and G_2. That means it does not exist $k \in \mathbb{Z}$ such that $n_2 = k \cdot q$. The motivation of imposing the condition that q does not divide n_2 is exposed in the next section. Practically, the STA chooses a random n_2 and divides it by q. If the result of the division is different from 0, the value of n_2 is kept. Else the STA increments n_2 by one and considers the result as the new value of n_2 ($n_2 := n_2 + 1$). The supplicant then ciphers the concatenation of n_1, n_2 and *pwd* using the Pub_{AS}. The result of the encryption is sent in message 3.
AS deciphers the received message using its private key $Priv_{AS}$. Then, the AS checks whether the STA's *pwd* is equal to the one it keeps in its password database. If the *pwd* is valid, the AS authenticates the STA, else the authentication process is stopped. Normally at this level

the authentication is finished but we extend it with the key derivation process as the AS is going to play the role of the PKG. As such, after STA authentication success, the AS computes the STA private key as $Priv_{STA} = s \times Pub_{STA}$ where s is the AS secret value and Pub_{STA} is the public key of the supplicant STA. Moreover, the AS encodes the computed $Priv_{STA}$ as $Enc_{Priv} = Priv_{STA} + n_2 sP$. Finally, the AS generates the message 4 which contains a random number n_3, the encoded private key Enc_{Priv} and an ID-based signature computed over the message formed by the concatenation of n_2, n_3 and the Enc_{Priv}.
After receiving message 4, the supplicant reconstructs the signed message by concatenating n_2, n_3 and the Enc_{Priv}. Then it verifies the message signature by the AS. If the signature verification is successful, the STA recovers its private key $Priv_{STA}$ from the Enc_{Priv} by doing the following operation: $Priv_{STA} = Enc_{Priv} - n_2 P_{pub}$. Note that $-n_2 P_{pub}$ is the inverse of $n_2 P_{pub}$. Then the supplicant uses its private key $Priv_{STA}$ to sign the random number n_3 and sends the message 5 back to the AS.
Upon receiving message 5, the AS verifies the signature using the public key of supplicant. If the signature is valid, the AS generates a final message to inform the authenticator of the authentication success. The AS may also register the STA identity, the value of n_2 and a timestamp in order to keep a trace of the authentication time and the private key validity period. At this stage, the authenticator deduces that the STA gets its private key and that the supplicant public key can be used to send some ciphered traffic to the supplicant.

B. Security Discussion

In this section, we present how the aforementioned authentication protocol avoids some attacks and how it can be enhanced to avoid other threats:
• *Denial of service attack (DoS):* To avoid that an attacker makes a DoS attack against the AS by sending a big amount of *Start-authentication* messages, we can limit the number of authentication requests to a certain threshold T_0. We suppose also that the authenticator is counting the number of requests sent by the supplicant STA. So when the number of requests relative to one STA exceeds this threshold, the authenticator must drop all the upcoming packets received from this STA for a certain period of time. In addition, the use of the authenticator signature helps the AS to authenticate the message origin and to be sure of its freshness thanks to the timestamp.
Using a threshold only at the authenticator level seems to avoid DoS attack but no Distributed DoS attack (DDoS). In fact, an attacker may control many STAs and launches a DDoS attack against the AS by sending different *Start-authentication* messages corresponding to the different controlled STAs (known as zombies). The aim of the attack is to flood the AS with a big amount of authentication requests. In order to avoid the attack, we limit the number of *Start-authentication* messages that an AS can receive from an authenticator during a time slot Δ_t to a threshold T_1. When the AS receives an *Start-authentication* message, it increments the counter of the received *Start-authentication* message related to the authenticator. If the counter value exceeds T_1, the AS drops the frames upcoming from this authenticator.
• *Replay attack:* The integers n_0, n_1, n_2 and n_3 are used to avoid replay attacks. We assume that these random numbers are at least 128 bit length. This implies that the probability of getting the same random number for two consecutive authentications is equal to $1/2^{128}$.
In the first message, n_0 arrives to the AS associated to the authenticator *timestamp* in order to make the authentication server verify the freshness of the session. Then in all upcoming messages, every random number is associated to the one that has been already sent in the previous message. For example in message 2, n_0 is associated to n_1 and in message 3, n_1 is associated to n_2. An attacker is able either to impersonate the STA or the AS, however he will not be able to replay old messages, unless nonces of an older authentication exchange correspond to the ones of the current exchange. For example, an attacker impersonating the STA, has to replay a previous message 3. However, for the replay to be successful, the value n_1 of this message 3 must correspond to the same value n_1 included in message 2. The probability of getting the same n_1 is equal to $1/2^{128}$ in case only one message 3 is known for replay by the attacker, but it increases in case the attacker knows more than one valid message 3.
• *Man in the middle attack:* The use of the secret password *pwd* in messages 2 and 3 makes the message forgery impossible. As such, when an attacker attempts a man in the middle attack, it needs to get the *pwd* in order to masquerade as the AS from one side and the supplicant STA from the other side. As the *pwd* is kept secret and sent ciphered, the attacker is not able to get it unless a brute force attack is performed.
• *Private key recovery from the Enc_{Priv}*: Concerning the private key derivation, an attacker may get the encoded private key of a supplicant but it has to find the secret n_2 in order to recover the private key of the supplicant. The problem of finding n_2 is equivalent to the discrete logarithm problem over an elliptic curve group [10].
Another possible security issue is linked to the encoded Enc_{Priv} that is transfered in cleartext and that could help attacker to generate valid signatures. That is why, we must verify whether an attacker may sign a message with the Enc_{Priv} and hopes to get the same signature as the one that could be computed with the true $Priv_{STA}$. Lets take the example of K.G. Paterson signature. We get the following results:

- when the $Priv_{STA}$ is used, the signature verification consists in comparing $\hat{e}(R,S)$ to $\hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{ID})^{H_3(R)}$. We have:
  $\hat{e}(R,S) = \hat{e}(kP, k^{-1}(H_2(M)P + H_3(R)Priv_{STA})$
  $\Rightarrow \hat{e}(R,S) = \hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P, Priv_{STA})^{H_3(R)}$
  $\Rightarrow \hat{e}(R,S) = \hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{STA})^{H_3(R)}$

- if the Enc_{Priv} replaces the $Priv_{STA}$ in the previous proof, we get the following result:
  $\hat{e}(R,S) = \hat{e}(kP, k^{-1}(H_2(M)P + H_3(R)Enc_{Priv}))$
  $\Rightarrow \hat{e}(R,S) = \hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{STA})^{H_3(R)} \cdot \hat{e}(P,P)^{n_2 s H_3(R)}$
  So we need to verify that $\hat{e}(P,P)^{n_2 s H_3(R)} \neq 1_{G_2}$ in order to avoid that an attacker uses the Enc_{Priv} to realize a signature collision. We have:
  $\hat{e}(P,P) \neq 1_{G_2}$
  $\Rightarrow \hat{e}(P,P) = g^t$ where g is the generator of G_2 and $0 < t < q$ where q is the order of G_2
  $\Rightarrow \hat{e}(P,P)^{n_2 s H_3(R)} = g^{t n_2 s H_3(R)}$
  We know that:
  1) $t < q$ and q prime $\Rightarrow t$ prime with q.
  2) $s \in \mathbb{Z}_q^* \Rightarrow s = s_1 + s_2.q$ where s_1 prime with q because $0 < s_1 < q$ and q prime.
  3) $H_3(R) \in \mathbb{Z}_q^* \Rightarrow H_3(R) = h_1 + h_2 q$ where h_1 prime with q because $0 < h_1 < q$ and q prime.
  4) n_2 is prime with q because q does not divide n_2 and q is a prime.

  $\Rightarrow t n_2 s_1 h_1$ is prime with q. Using the aforementioned information, we get the following result:
  $\Rightarrow g^{t n_2 s H_3(R)} = g^{t n_2 \times (s_1 + s_2.q) \times (h_1 + h_2 q)}$
  $\Rightarrow g^{t n_2 s H_3(R)} = g^{t n_2 s_1 h_1 + q \times t n_2 (s_1 h_2 + s_2 h_1 + s_2 h_2 q)}$
  $\Rightarrow g^{t n_2 s H_3(R)} = g^{t n_2 s_1 h_1}$ and $t n_2 s_1 h_1$ is prime with q
  $\Rightarrow \hat{e}(P,P)^{n_2 s H_3(R)} = g^{t n_2 s H_3(R)} = g^{t n_2 s_1 h_1} \neq 1_{G_2}$
  This shows that the two signatures generated by the Enc_{Priv} and $Priv_{STA}$ can never be equal. Consequently, an attacker can not use Enc_{Priv} to generate the same signature as with $Priv_{STA}$.

V. Extensions For Maintaining Security Throughout The Session

A. Updating Public Elements and Password

The *public elements* and the secret value s used by the AS for private key generation must be renewed periodically to avoid attacks like brute force attack. The new *public elements* and the new private key are sent from the AS to a STA ciphered by the current STA public key (Figure 4). The message may contain a timer to indicate to the STA when the current private/public keys must be definitely replaced by the new ones.

When the AS decides to change the *public elements*, it sends a *Change-public-elements* message to the STA. This message includes a nonce (n_1), the STA new private key ($new - Priv_{STA}$), the new *public elements*, a timer to indicate to the STA when to replace the current keys by the new ones and the *pwd* shared with this STA. All these aforementioned elements are ciphered by the STA current public key.

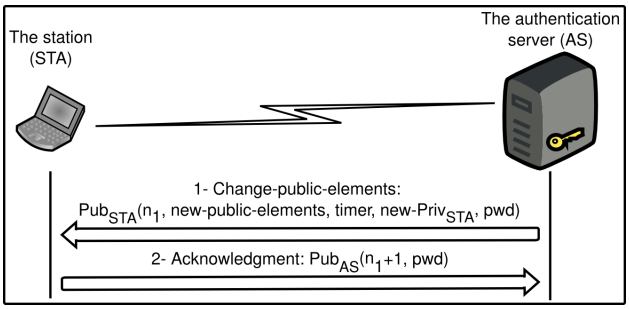

Fig. 4. The renewal of *public elements*.

The STA deciphers the received *Change-public-elements* message, and it gets the new parameters including its $new - Priv_{STA}$. The STA does not use these parameters until the expiration of the timer which indicates the end of the validity of the current key pair. The STA sends back an *Acknowledgment* message which contains the n_1 incremented by 1 and the secret *pwd*. The incrementation of the n_1 proves to the AS that the message is sent in response to message 1.

The *pwd* used between the AS and the supplicant may also be changed after a certain number of successful authentications. The two STAs may use a signature based Diffie-Hellman algorithm to exchange a new *pwd*.

In the case that the STA does not want to pick an n_2 that is not a multiple of q, a secure Diffie-Hellman (DH) scheme can be used instead to establish a new secret key between the supplicant and the AS. That means that the STA does no longer need to verify that q does not divide n_2. Then the secret DH key serves to transport the private key $Priv_{STA}$ from the AS to the supplicant STA. So, this scheme is an alternative to encoding the $Priv_{STA}$ in Enc_{Priv}.

As illustrated in Figure 5, the AS appends its part of the DH key (k_{AS}) to message 2. The k_{AS} is included also in the signature in order to avoid that an attacker attempts a man in the middle attack against DH. Upon receiving message 2, the supplicant STA first authenticates the AS. Then it generates its part of the DH key (k_{STA}) and computes the final DH key (k_{DH}). In addition, the supplicant includes this k_{STA} in message 3 in order to authorize the AS to generate the final k_{DH}. The k_{STA} is sent to the AS enciphered with Pub_{AS}. The AS deciphers the received message 3, then it authenticates the STA using the received *pwd*. If the authentication is successful, the AS generates the DH key k_{DH} using the received k_{STA}.

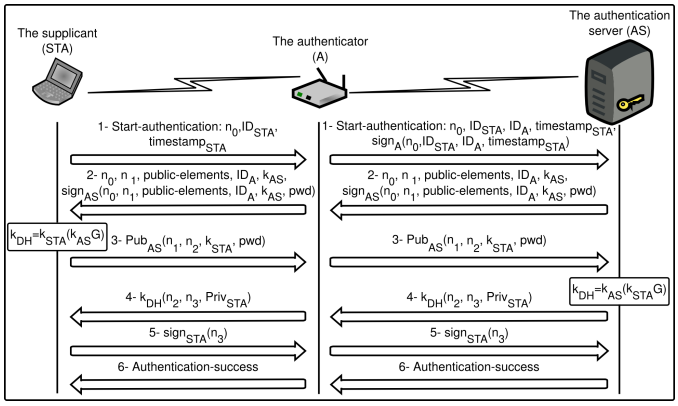

Fig. 5. ID-based authentication with DH for $Priv_{STA}$ secure transfer.

The AS next computes $Priv_{STA}$ and sends it to its owner enciphered with k_{DH}.

B. STA Subsequent Authentication

After the initial authentication, each pair of STAs can mutually authenticate using a signature scheme. An authentication scheme can be defined as presented in Figure 6. The authentication starts when a STA (STA1) sends a *Start-authentication* message to another STA (STA2). This request includes a random challenge (*challenge1*) and a timestamp ciphered with the public key of the STA2 (Pub_{STA2}) . Upon receiving the request message, STA2 deciphers the content using its private key ($Priv_{STA2}$). It then checks the timestamp value to detect replays. Then, STA2 sends a message to STA1 which contains the value of *challenge1*, a new *challenge2* and a signature over the concatenation of the *challenge1*, the *challenge2* and the *public-elements* that STA2 is using. The objective of sending the *public-elements* used by STA2 to STA1 is for STA1 to check that the same parameters applies on STA2. That means indirectly that STA2 has been initially authenticated by the same AS as STA1 did. In addition, that implies that the STA2 *public-elements* has not been forged by an attacker.
The STA1 reconstructs the message that has been signed by STA2 by concatenating the values of *challenge1* and the *challenge2* to its *public-elements* which have been received from the AS. Then, the STA1 verifies the signature of the constructed message. If the signature is invalid, the STA1 stops the authentication operation. Actually, a signature verification failure means that either one of the signed challenges is wrong or the STA2 *public-elements* are different from STA1 *public-elements*. That implies in the two cases that STA2 has been attacked. If the signature verification is successful, the STA1 sends to the STA2 the third message which contains the signature of the *challenge2* and its *public-elements*. The STA2 verifies the signature received from the STA1 within the message. If the signature is valid, the STA2 sends an *Authentication success* message to the STA1.

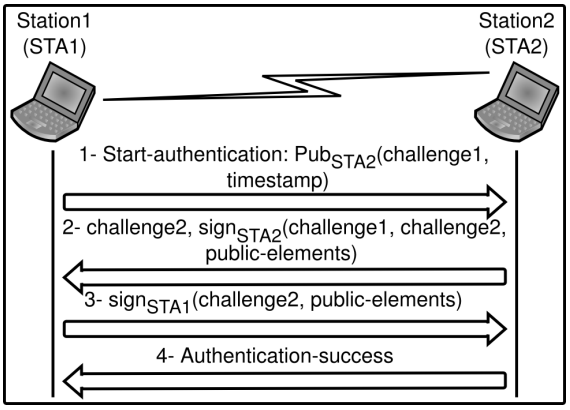

Fig. 6. ID-based authentication scheme.

VI. ID-based Authentication Scheme in IEEE 802.11s Environment

The 802.11s architecture defines a new entity - the Mesh Key Distributor (MKD) - which controls the key derivation. The MKD is in charge of deriving the keys needed for the 4-way handshake between the supplicant mesh STA and the mesh authenticator. At the end of the supplicant authentication, the MKD receives from the AS the information needed for the key derivation. The MKD then computes the key that the mesh authenticator (MA) needs in order to perform the 4-way handshake with the supplicant STA. In fact, the MKDs are used to distribute the key derivation operation that can be consuming in terms of power for the AS. The IEEE 802.11s standard assumes that there is a security association between the MKD and the AS.
In this section, we show how our proposed authentication scheme can be applied to the 802.11s mesh architecture. Note that the MKD purpose is similar to the PKG function. As such, the MKD can better serve for key derivation than the AS, and the AS is assumed to trust the MKD for the key derivation operation. Meanwhile the AS chooses the *public elements* and the secret value s which are used during the private key derivation operation. In fact, when the AS chooses the *public elements*, it keeps its hand in the network control and management.
We decided to enhance the *public elements* with a new parameter called P_{AS}. P_{AS} is an elliptic curve point computed as $P_{AS} = s_{AS} \cdot P$ where s_{AS} is a secret value selected locally by the AS such that s_{AS} is different from s. The point P_{AS} must be different from the point P_{pub}. The aim of using P_{AS} is to avoid that an MKD impersonates as the AS by signing messages using the AS private key that the MKD computes as the product of the AS public key and the secret s received from the AS. In fact, the MKD will receive the secret s from the AS in order to compute the supplicant STA keys. Consequently, the MKD is able to compute the private key of the AS knowing the AS identity and s. To avoid this problem, we supposed that the AS will use the secret s_{AS} to compute its own key pair and the secret s will be used for the computation of the remaining stations key pairs. Moreover, we have introduced P_{AS} to replace P_{pub} when verifying a signature that comes from the AS. If we use K.G. Paterson signature scheme for example, all the signature verification will consist on comparing $\hat{e}(R,S)$ to $\hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{ID})^{H_3(R)}$. While the AS signature verification consists in comparing $\hat{e}(R,S)$ to $\hat{e}(P,P)^{H_2(M)} \cdot \hat{e}(P_{AS}, Pub_{ID})^{H_3(R)}$.
The Figure 7 presents how our ID-based authentication scheme can be adapted to the 802.11s mesh network architecture.
When a mesh STA joins the network, it receives Beacon frames from its one-hop neighbors. The STA then selects one of its neighbor as the mesh authenticator (MA). In addition, it synchronizes its clock in accordance to its neighbors clocks. Then, it starts the authentication by sending a *Start-authentication* message to the MA.

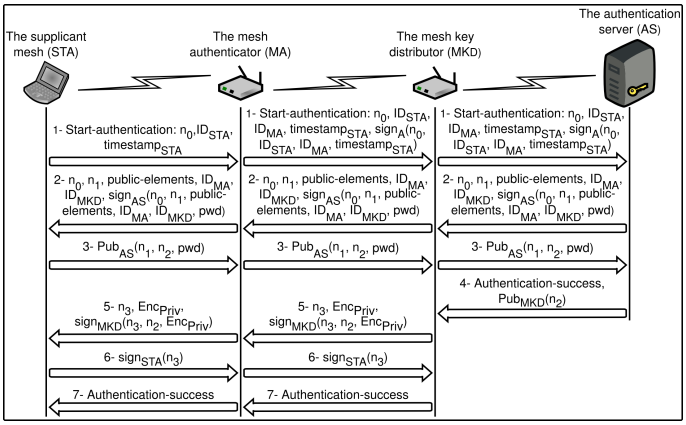

Fig. 7. ID-based authentication scheme for 802.11s mesh network.

The MA appends its identity (ID_{MA}) to the received message 1 and signs it before transmitting it to the MKD. The signature is used to authenticate the MA to the AS and to avoid DoS attacks. Each MA is connected to an MKD. The MKD acts as a pass-through station for the authentication message until receiving message 4 from the AS.

Upon receiving the *Start-authentication* message from the supplicant STA, AS checks the STA timestamp value. Then, AS generates message 2 which includes a signature of the *public-elements*, the random numbers used to avoid the replay attack (n_0 and n_1), the identity of the MA (ID_{MA}) and the identity of the MKD (ID_{MKD}) which is the only new field compared to the aforementioned initial authentication scheme. If the supplicant STA succeeds to verify the message 2 signature, it gets an authenticated MKD identity. This ID_{MKD} authentication is necessary because the supplicant STA uses this identity to compute the MKD public key (Pub_{MKD}) needed for the message 5 signature verification.

After the authentication of the AS through the verification of the message 2 signature, the supplicant STA generates the message 3 in order to authenticate itself to the AS. After receiving the message 3, the AS authenticates the supplicant STA by comparing the received *pwd* to the one contained in its passwords database. If the authentication is successful, the AS sends the message 4 to the MKD to launch the computation of the supplicant STA private key. The message 4 contains the random n_2 chosen by the supplicant STA such that q does not divide it. This n_2 is sent ciphered by the MKD public key (Pub_{MKD}). The MKD uses n_2 to encode the supplicant private key ($Priv_{STA}$) into Enc_{Priv}.

The authentication scheme ends when the MKD receives the supplicant STA signature of the random n_3. At this moment, the MKD transmits the *Authentication success* message to the MA which is relayed to the supplicant STA. For the subsequent authentication with other mesh STAs, the supplicant STA can use a public/private key authentication scheme as presented in the previous section. In addition, the two STA can include an authenticated DH key exchange using a signature mechanism in order to share a *Pairwise Transient Key (PTK)* [8]. Moreover, each STA has to control the transmission of its *Group Temporal Key (GTK)* for its peers. The *GTK* is a key that serves to encrypt the broadcast/multicast traffic of the STA.

VII. Conclusion

In this paper, we presented a new authentication scheme that relies on the ID-based cryptography. Note that ID-based cryptography was initially introduced to replace the PKI deployment and alleviate the burden of the certificate management. As such, it is really suitable to wireless networks like ad hoc networks and mesh networks where nodes are resource constrained. To closely match the IEEE 802.11s mesh network needs, we adapted our authentication scheme to the IEEE 802.11s mesh architecture by assigning a specific role to the Mesh Key Distributor (MKD). Our future works are focusing on validating the proposed protocol and testing its performance in terms of time and computation power. Also evaluation of its scalability and ease of deployment is part of our perspectives.

References

[1] *IEEE Std 802.1X-2004: Port Based Network Access Control*, IEEE Standard, dec 2004.

[2] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748 (Proposed Standard), Internet Engineering Task Force, Jun. 2004, updated by RFC 5247. [Online]. Available: http://www.ietf.org/rfc/rfc3748.txt

[3] K. G. Paterson and G. Price, "A comparison between traditional public key infrastructures and identity-based cryptography," Royal Holloway University of London, Tech. Rep., 2003.

[4] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.

[5] K. G. Paterson, "Id-based signatures from pairings on elliptic curves," *Electronics Letters*, vol. 38, no. 18, pp. 1025 – 1026, 29 2002.

[6] F. Hess, "Efficient identity based signature schemes based on pairings," in *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*. London, UK: Springer-Verlag, 2003, pp. 310–324.

[7] L. Wenju, S. Yuzhen, and W. Ze, "A wireless mesh network authentication method based on identity based signature," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, 24-26 2009, pp. 1 –4.

[8] *IEEE P802.11s/D2.06: Part 11: Wireless LAN MAC and Physical layer specifications. Amendment 10: Mesh networking*, IEEE Working Draft Proposed Standard, Rev. 2.06, jan 2009.

[9] *IEEE Std 802.11-2007: Part 11: Wireless LAN MAC and Physical layer specifications*, IEEE Standard, Rev. Revision of IEEE Std 802.11-1999, jun 2007.

[10] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.